PROBLEMI INTRATTABILI

I problemi intrattabili sono problemi che possono essere risolti ma non in tempi accettabili *(tempo esponenziale)*.>
Spesso sono problemi simili a quelli risolvibili, ma hanno una piccola differenza che li fa cadere nei problemi intrattabili. Ovviamente, i problemi intrattabili non sono ***sempre*** non risolvibili facilmente, dipende dall'input.

Si pensa che gli algoritmi per i problemi intrattabili non esistano; non si è riusciti a dimostrare che non esista un tale algoritmo.

P ?= NP
Dove P = problemi che sappiamo risolvere in tempi rapidi, NP non sappiamo se possiamo farlo.
Alcuni problemi sono dimostrabilmente intrattabili, ma solo alcuni.
NP quindi sono problemi per il quali abbiamo il forte sospetto che non possano essere risolti in tempo polinomiale, ma non è dimostrabile.

Formalmente queste sono classi di linguaggi; i problemi si associano ai linguaggi.
In particolare, ai linguaggi si assegna la versione di decisione dei problemi; se non possiamo risolvere quella certamente non possiamo risolvere la versione di ottimo.
$L_\pi = \{x | x \text{ è } codifica\ di\ un'istanza\ con\ risposta\ Y\}$

Definiamo la classe P = {L | L è deciso tramite una MDT da un algoritmo in tempo O(p(n))} con n = |X|
P quindi è la classe dei problemi di decisione risolti da un algoritmo in tempo polinomiale.

Cosa succede per quelli che non riusciamo a classificare in P?
Solitamente li classifichiamo NP.

VERIFICA DI UNA SOLUZIONE IN TEMPO POLINOMIALE
Qualcuno ci fornisce una possibile soluzione con un ***certificato***; se esiste ci da la risposta esatta, altrimenti ci da una risposta sbagliata per farci capire che non esiste.
Quanto tempo ci mettiamo a verificare che il certificato ci dia risposta Y or N?

NP = {Problemi di decisione verificabili in tempo O(p(n))} quindi
NP = {L | L è verificato in O(p(n)) da un algoritmo}
NP sta per non deterministic polynomial

Ma chi ci da il certificato? Alcuni problemi sono verificabili in tempo polinomiale, ma non sappiamo come ottenere il certificato.

LEGAME CLASSE P E NP
Certamente $P \subseteq NP$; un problema risolvibile in tempo polinomiale è certamente verificabile in tempo polinomiale.
Ma $P \subset NP$ o $P = NP$? Non sappiamo dirlo. Si pensa la prima, ma non se ne ha la certezza.

I problemi in NP non hanno tutti la stessa difficoltà, alcuni sono più complessi. Facciamo quindi un ordinamento parziale basato sulla loro difficoltà.

Riduzione polinomiale: se trasformo l'istanza di un problema A in un'istanza di un problema B polinomiale in tempo polinomiale, posso dire che A è risolvibile in tempo polinomiale e che B è più difficile di A, perché una volta che rispondo a B, rispondo anche ad A.

Quindi
$\pi_1 < \pi_2$ in NP   se un giorno dimostro che $\pi_2 \in P \implies \pi_1 \in P$
se un giorno dimostro che $\pi_1 \notin P \implies \pi_2 \notin P$

Cos'è un problema NP Completo?
Un problema tale per cui

1. $\pi \notin NP$
2. $\forall \pi' \in NP, \pi' < \pi$

Gli NP Completi sono i più difficili in NP.

Se $\exists \pi \in NPC \cap P \implies P = NP$, per il teorema di prima; dato che i NPC si riducono a vicenda, se riesco a risolverne anche uno solo in tempo polinomiale, li risolvo tutti. Praticamente se riesco a risolvere TSP in tempo polinomiale, posso anche decifrare un file criptato in tempo polinomiale. Per questo si pensa che NP non sia uguale a P.

Se $\exists \pi \in NP - P \implies \forall \pi' \in NPC, \pi' \notin P$, visto che tutti quelli in NP si riducono in NPC.

Il primo problema NPC è SAT *(soddisfacibilità)*, se dimostriamo che quello è NPC. È stato dimostrato essere NPC.
Ora basta vedere che SAT si riduce agli altri sottoproblemi, quindi anche gli altri sono NPC.

Un problema NP_Hard è un problema difficile almeno tanto quanto risolvere un NP.